Amadey

Technical Analysis Report

ZAYOTEM
ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Contents

Overview

Amadey is a botnet that appeared around October 2018. Periodically sends information about the system and installed AV software to its C2 server and polls to receive orders from it. Its main functionality is that it can load other payloads (called “tasks”) for al lor specifically targeted computers compromised by the malware.

Some of the information obtained from infected devices include;

- Registry manipulation,
- Device properties and file information,
- Security applications,
- Operating system information

Nbveek.exe Analysis

Name	Nbveek.exe
MD5	77e0a0a90e0231493bd421f4cdab0668
SHA256	75520c76a4051b2be15db8625f35d4c1c63d93686bf849e6fc67f4e62d2fd000
File Type	PE32 / EXE

Static Analysis

At the first glance, base64 values stand out in the malicious file, except fort he APIs used in strings.

```
.rdata:00433430  00000011  C  KQKkIcN JIhBGC==
.rdata:00433444  00000011  C  9gy4UM0wHo06SS==
.rdata:00433458  0000000D  C  RUCKQuKYO3R=
.rdata:00433468  00000029  C  IUCAUMK5VUFxN2DdTYaKNWMHMyZJKABAFwYWO9F=
.rdata:00433494  00000009  C  FwYWP9Fn
.rdata:004334A0  00000009  C  FcpxM7==
.rdata:004334B0  00000045  C  RUYIQviGQm0eMYcgcqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eYjLqK46mWK==
.rdata:004334F8  0000005D  C  RUYIQviGQm0eMYcgcqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eVT8s3I BWRyeQNSq9kGV...
.rdata:00433558  0000000D  C  RXGj8dW69C==
.rdata:00433570  00000019  C  VWQmDsdIEHOPKGcPIGYvD4zi
.rdata:0043358C  0000003D  C  RUYIQviGQm0eMYcgcqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eYjLq
.rdata:004335CC  0000000D  C  hKwUwCxIBNi
.rdata:004335DC  00000015  C  FwYGUMCq IZiE3ILIGx=
.rdata:004335F4  0000000D  C  RByxUTOm8ZR=
.rdata:00433608  00000059  C  RUYIQviGQm0eMYcgcqYv43U21B4I3oSyaXCeLT0D9o0w7HQicrCI430eVT8s3I BWRyePSmq8IBiK...
.rdata:00433664  00000015  C  GPKVMLOVQmdILWoCJU==
.rdata:0043367C  00000009  C  Uyuy8q==
```

Image 1- IDA strings

When some of the values that are considered to be Base64 are decoded, meaningless values such as, “**)p^G\ė=” , “"„黩Q{+d*” , “&#” , “)!IF**” appear. These values are made sense at runtime using a specified key.

```
v25 = (const CHAR *)lpFileName;
if ( v60 >= 0x10 )
  v25 = lpFileName[0];
v26 = GetFileAttributesA(v25);
if ( v26 == -1 || (v26 & 0x10) == 0 )
{
  v27 = (const CHAR *)lpFileName;
  v45 = 0;
  if ( v60 >= 0x10 )
    v27 = lpFileName[0];
  CreateDirectoryA(v27, (LPSECURITY_ATTRIBUTES)v45);
}
v28 = (const CHAR *)lpFileName;
if ( v60 >= 0x10 )
  v28 = lpFileName[0];
v29 = GetFileAttributesA(v28);
if ( v29 != -1 && (v29 & 0x10) != 0 )
{
  v44 = (LPCSTR)sub_415850(FileName);
  v37 = (const CHAR *)sub_415850(v61);
  CopyFileA(v37, v44, 1);
  if ( (unsigned __int8)sub_405020((char *)FileName) )
  {
    v58 = &v40;
```

Image 2- IDA pseudo code

Here, it is seen that by searching for the folder in a directory, it is **created** if it does not exist, and then searching for a file under that folder and if not, the **handheld file** is **copied into it**.

```
:00403A4A push    offset a1       ; "1"
:00403A4F call    sub_416B10      ; Call Procedure
:00403A54 add     esp, 8          ; Add
:00403A57 xor     edx, edx        ; Logical Exclusive OR
:00403A59 test    al, al          ; Logical Compare
:00403A5B mov     ecx, 5
:00403A60 cmovz   ecx, edx        ; Move if Zero (ZF=1)
:00403A63 mov     edx, offset aRunas ; "runas"
:00403A68 push    ecx             ; Size
:00403A69 mov     ecx, offset Src
:00403A6E cmovnz  ecx, edx        ; Move if Not Zero (ZF=0)
:00403A71 push    ecx             ; Src
:00403A72 lea     ecx, [ebp+lpOperation] ; void *
:00403A75 call    sub_415CD0      ; Call Procedure
:00403A7A cmp     [ebp+arg_44], 10h ; Compare Two Operands
:00403A7E lea     edx, [ebp+lpParameters] ; Load Effective Address
:00403A81 push    0               ; nShowCmd
:00403A83 cmovnb  edx, [ebp+lpParameters] ; Move if Not Below (CF=0)
:00403A87 lea     ecx, [ebp+lpFile] ; Load Effective Address
:00403A8A cmp     [ebp+arg_2C], 10h ; Compare Two Operands
:00403A8E lea     eax, [ebp+lpOperation] ; Load Effective Address
:00403A91 push    0               ; lpDirectory
:00403A93 cmovnb  ecx, [ebp+lpFile] ; Move if Not Below (CF=0)
:00403A97 cmp     [ebp+arg_14], 10h ; Compare Two Operands
:00403A9B push    edx             ; lpParameters
:00403A9C cmovnb  eax, [ebp+lpOperation] ; Move if Not Below (CF=0)
:00403AA0 push    ecx             ; lpFile
:00403AA1 push    eax             ; lpOperation
:00403AA2 push    0               ; hwnd
:00403AA4 call    ds:ShellExecuteA ; Indirect Call Near Procedure
```

Image 3- IDA view

It is clearly seen that the copied file is run with the "**runas**" parameter right after the file copy operation.

Dynamic Analysis

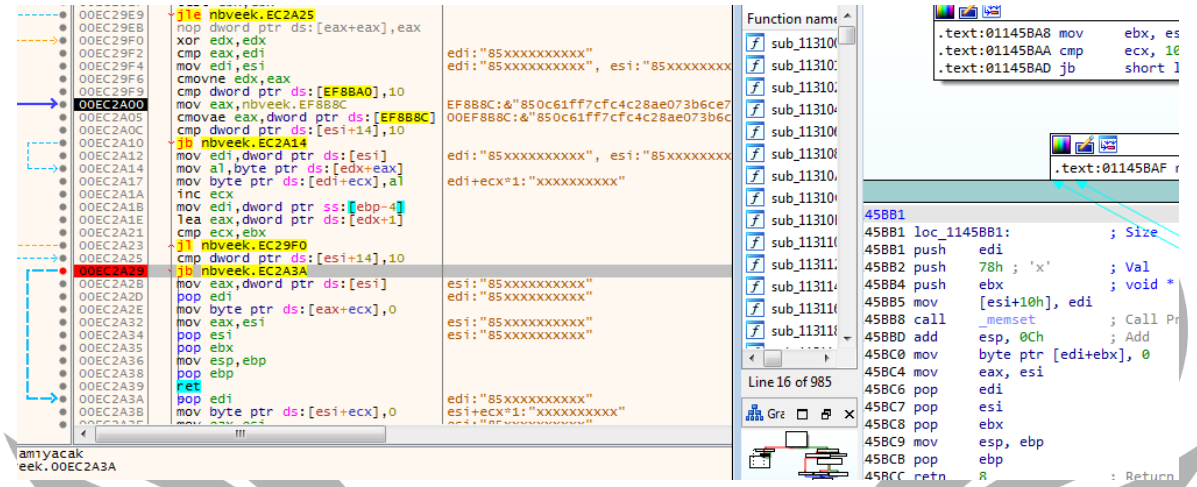

Image 3- x32dbg

The key value is set to the **size of the value to be resolved**.

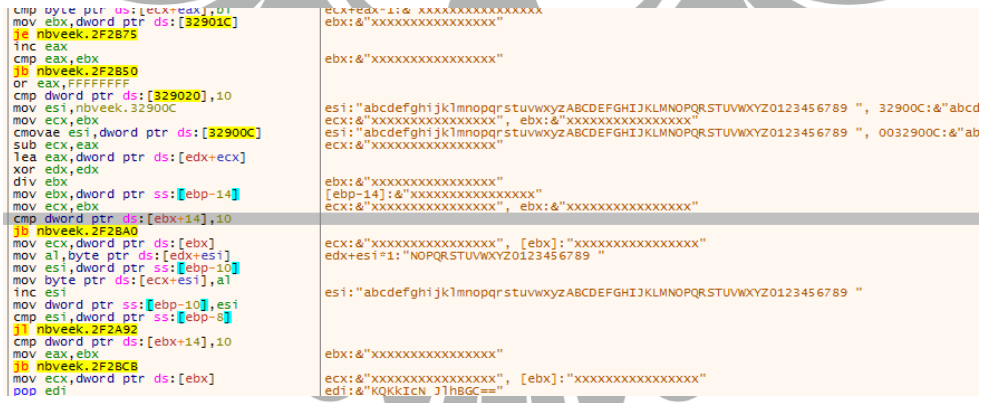

Image 4- x32dbg string parsing operation

The value to be solved with the obtained value creates **meaningful base64 values** after a number of operations.

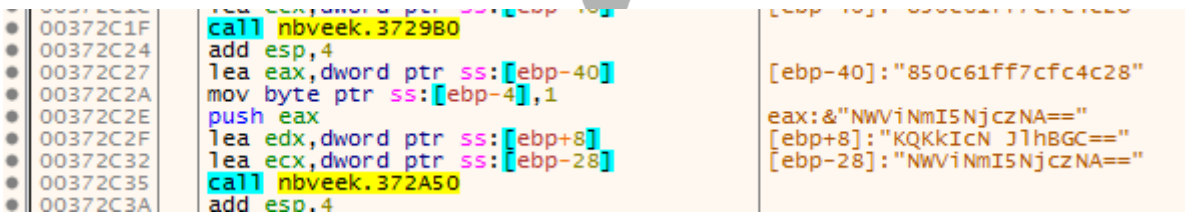

Image 5- x32dbg parsed example

KQKkIcN JlhBGC== // NWViNmI5NjczNA== // **5eb6b96734 (Directory name)**

Image 6- AntiDebug

It gets the working file location using **GetModuleFileNameA** Api. Compares it with the place where, copied itself before. If it is not working in the same place, it closes itself.

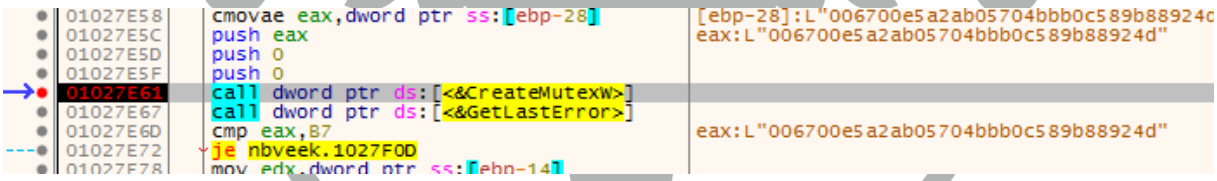

Image 7- x32dbg Mutex

A **mutex** with the name “006700e5a2ab05704bbb0c589b88924d” is created and “ERROR_ALREADY_EXISTS” (0x**B7**) is checked with **GetLastError**. If the mutex is created, it is assumed that it is **already running** and the program is closed.

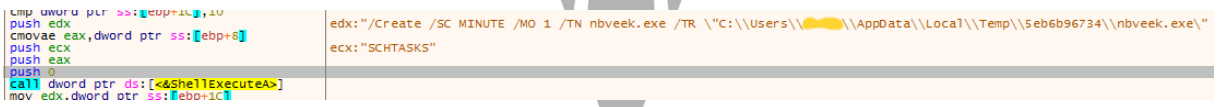

Image 8- Cmd SCHTASKS persistence

With the script in the image, it runs the program **at the beginning of every minute** and ignores the warnings with the **"/F" (Force)** parameter.

```
"kodex"
":N\""
"&&"
"CACLS \""
"nbveek.exe"
"\" /P \""
"kodex"
":R\" /E"
"&&"
"echo Y|CACLS \""
"..\\5eb6b96734"
"\" /P \""
"kodex"
":N\""
"&&"
"CACLS \""
"..\\5eb6b96734"
"\" /P \""
"kodex"
":R\" /E"
"&&Exit"
```

Image 9- Cmd script

The bitwise deciphered values are **stored in the stack** and combined before being executed.

```
"C:\Windows\System32\cmd.exe" /k echo Y|CACLS "nbveek.exe" /P
"kodex:N"&&CACLS "nbveek.exe" /P "kodex:R" /E&&echo Y|CACLS
"..\5eb6b96734" /P "kodex:N"&&CACLS "..\5eb6b96734" /P "kodex:R"
/E&&Exit
```

*Kodex = %USERNAME%

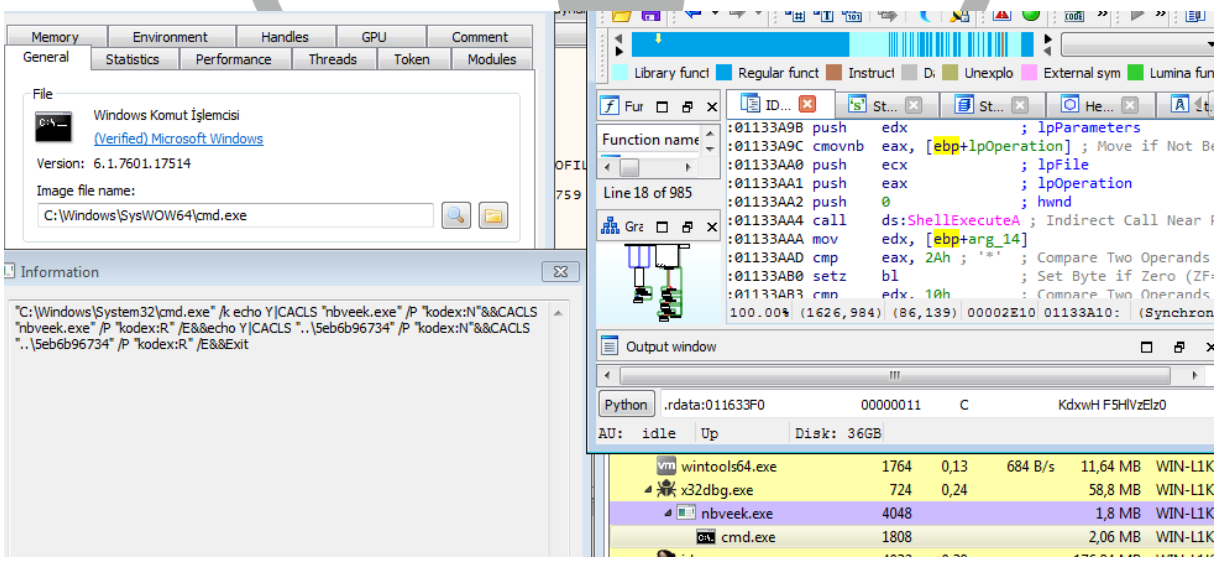

Image 10- Cmd script_2

This script running in “cmd.exe” allows to check the **access list**.”echo Y|” with the command, “Are you sure?” It provide the answer to the question to be entered as "YES". “**Kodex**:” provides to enter the answers that the user must enter, in here firstly “**None**” permission and then “**Read**” permission is given to the “**nbveek.exe**” file. Then the same process is applied for the folder where the **malware is located**.

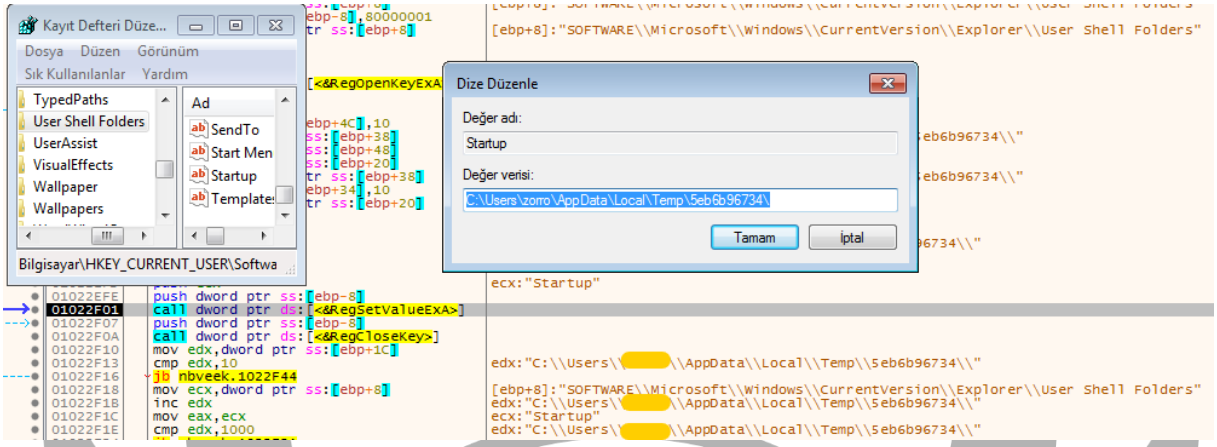

Image 11- Registry persistence

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\
User Shell Folders

The **default value** of this path in the registry is the following file path.

"%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup"

However, with this process, the malware sets the **default startup path** to its own directory.

```
201     v10 = v124.dwMajorVersion;
202     if ( v124.dwMajorVersion == 10 )
203     {
204       if ( !v124.dwMinorVersion )
205       {
206         v11 = (v125 != 1) + 1;
207 LABEL_86:
208         dword_438930 = v11;
209       }
210       return dword_438930;
211     }
212     if ( v124.dwMajorVersion != 6 )
213       goto LABEL_76;
214     v12 = v124.dwMinorVersion;
215     if ( v124.dwMinorVersion == 3 )
216     {
217       v11 = (v125 != 1) + 3;
218       goto LABEL_86;
219     }
220     if ( v124.dwMinorVersion != 2 )
221       goto LABEL_72;
222     if ( v125 == 1 )
223     {
224       v123 = &v105;
225       dword_438930 = 1;
226       sub_415970(&dword_439054);
227       sub_402BE0(v61, v68, v75, v82, v89, v97);
228       v131 = 1;
229       sub_415970(&dword_439264);
230       sub_402BE0(v42, v45, v48, v51, v54, v57);
231       v131 = -1;
232       v13 = (_DWORD *)REGOPEN(
233                         v118,
234                         HKEY_LOCAL_MACHINE,
235                         v62,
```

Image 12- Version check

Version control is carried out within the "**OSVERSIONINFOEXA**" structure, which is stored for later logging. If the operating system is **Windows Server 2012 or Windows 8**, it is controlled via registry. It also searches for **2016, 2019, 2022** values in the **ProductName** value **in SOFTWARE\Microsoft\Windows NT\CurrentVersion** according to the data checked with the **GetNativeSystemInfo** API. According to these results, "**OS**" logging changes.

Image 13- AV check

The presence of selected security software under the **ProgramData** folder is checked. If there is a folder **(even empty)** created with the name of any of the **security applications** specified under the ProgramData folder, it detects it as "antivirus exists". (&av=1)

AVAST Software	Avira	Kaspersky Lab	ESET
Panda Security	Doctor Web	AVG	360TotalSecurity
Bitdefender	Norton	Sophos	Comod

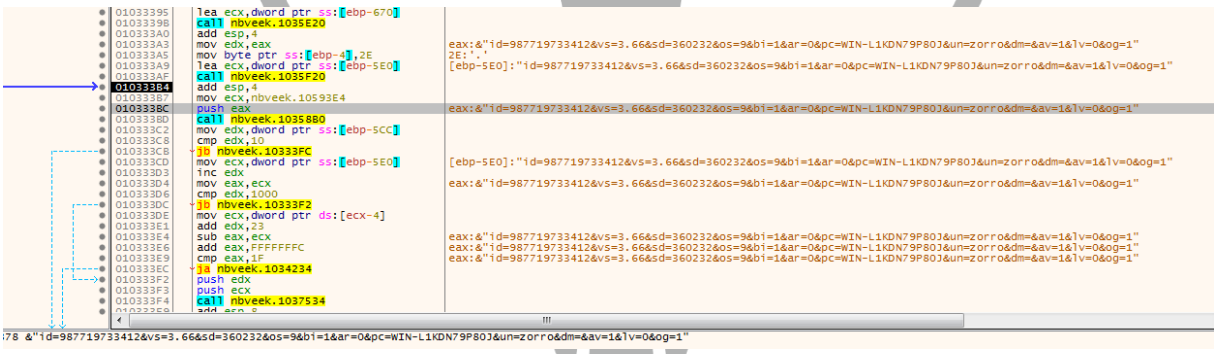

Image 14- LSAPolicyLookup

In the image above, the information collected about the system is combined into a single text to be sent to the **command and control** server.

Image 15- Harmful URL

It tries to open the URL in the image and **read the file**, but cannot do this because it is **closed**.

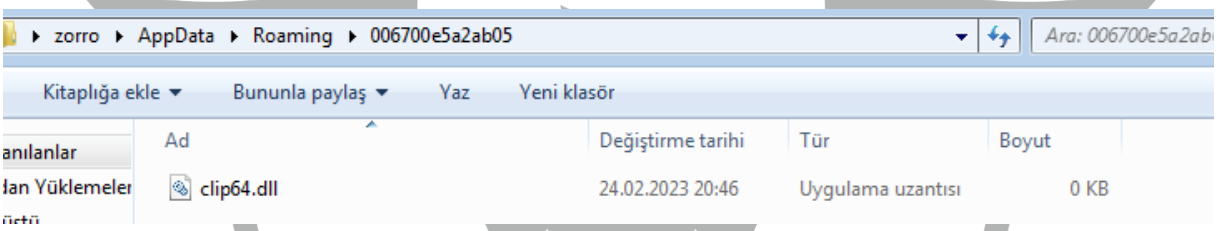

Image 16- File to download

This is the directory of the file that is expected to be written after being read. Previously, the **Mutex** is created here.

```
C:\Users\%USERNAME%\AppData\Roaming\006700e5a2ab05\
```

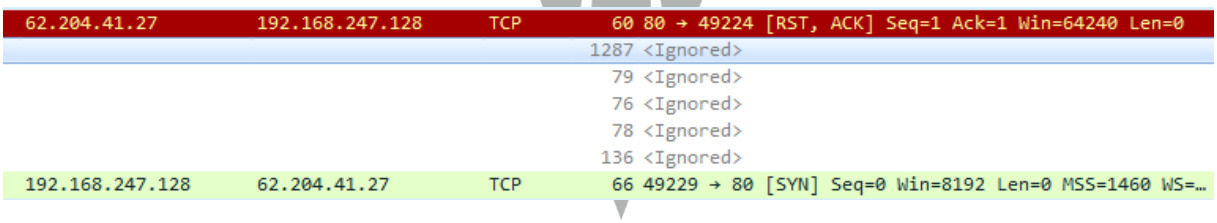

Image 17- Wireshark view

With the created **thread**, the program sends a **TCP** packet to the command and control server each time. However, since the server is down, any operation does not continue because **RST packets** are received.

```
end = 0x433E59      # Here is for end of the encoded string's
start = 0x4333F0    # Start of encoded string's address

value = idaapi.get_bytes(start,end-start)
stringValue = value.decode("utf-8")
listedValue = stringValue.split('\x00')

def yaz(final):
        if(final != ""):
                text_file = open("Decrypted.txt", "a")
                text_file.write(final+"\n")
                text_file.close()

def decodeToBase64(listedValue):
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 "
    key = "850c61ff7cfc4c28ae073b6ce7172cbd850c61ff7cfc4c28ae073b6ce7172cbd"
    alphabet1 = alphabet[::-1]

    for line in listedValue:
        final = ""
        counter=0
        while(len(line)!= counter and line[counter]!="=" and line[counter]!="+"):
            indexAlp= 0
            while(alphabet1[indexAlp] != key[counter]):
                indexAlp+=1
            indexAlp2 = 0
            #print(line[counter])
            while(alphabet[indexAlp2] != line[counter]):
                indexAlp2+=1

            final +=alphabet[((indexAlp+1) + (indexAlp2+1)) % 63 -1]
            counter +=1
        yaz(final)
decodeToBase64(listedValue)
```

The **ciphertexts** are written to the text file with the above **IdaPython** code converted to **Base64** values.

YARA Rule

```
import "pe"

rule Amadey{

meta:

        author="enessakircolak"

        date= "01.03.2023"

strings:

        $a = "Amadey.pdb"

        $b = {83 3D ?? ?? ?? ?? 10 BE ?? ?? ?? ?? 8B CB 0F 43 35 ?? ?? ?? ?? 2B C8 8D 04 0A 33 D2 F7 F3 8B 5D EC 8B CB 83 7B 14 10 72 02 8B 0B 8A 04 32 8B 75 F0 88 04 31 46 89 75 F0 3B 75 F8}

        $mutex = "006700e5a2ab05704bbb0c589b88924d"

        $key = "850c61ff7cfc4c28ae073b6ce7172cbd"

        $enc1 = "KdxwH F5HlVzElz0"

        $enc2 = "RUYIQviGQm0eMYcgcqYv43U21B4l3oSyaXCeLT0D9o0w7HQicrCl430eVT8s3I BWRyeQNSq9kGV4IMpbGqC43sm3Tzv"

        $enc3 = "OWK2OcK57Z4nN5cwdKKpKX0ofs=="

        $enc4 = "AKeF7S 5VY 2EWImc7qr53g2eSZqFkCp9XyvGMWmIJ9BIWeb0J5AnMjhCueFACp8Qmn7cKyVV5k"

        $api1 = "CreateMutex"

        $api2 = "GetVersionEx"
```

```
        $api3 = "CreateThread"
        $api4 = "ShellExecute"
        $api5 = "HttpOpenRequest"
        $api6 = "InternetOpenUrl"
        $api7 = "CopyFile"
        $api8 = "LoadLibraryEx"
        $api9 = "CreateDirectory"
        $api10 = "RemoveDirectory"
        $api11 = "GetFileAttributes"
        $api12 = "RegCloseKey"

condition:
        uint16(0) == 0x5a4d
        and filesize <= 1MB
        and pe.imports("WININET.dll")
        and(
        any of ($a,$b,$api*) or all of ($enc*,$mutex,$key)
        )
}
```

MITRE ATTACK TABLE

Reconnaissance	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	C&C	Exfliration
Gather Victim Host Information (T1592)	**Windows Command Shell (T1059)**	**Scheduled Tasks/Job (T1053)**	**Manipulate System Process (T1053)**	**Obfuscated Files or Information (T1027)**	**OS Credential Dumping (T1003)**	**Remote Acces Software (T1219)**	Scheduled Transfer (T1029)
	Scheduled Task (T1053)	**Startup Folder (T1547)**	**Registery Run Keys (T1547)**	**Modify Registry (T1112)**	**Credentials in Registry (T1552.002)**		
	Startup Folder (T1547)						

Solution Proposals

1. The system should be kept up to date.
2. Every process must be audited at runtime.
3. Reliable anti-virus software should be used (even if not used, the file with that name should be located under ProgramData).
4. All documents should be used after inspected.

Prepared By

Enes Şakir ÇOLAK LinkedIn